Electronic design:
a new field of investigation for large scale optimization

Marc Sevaux * André Rossi * Kenneth Sörensen †

* University of South Brittany – UEB
Lab-STICC, Centre de Recherche, F-56321 Lorient France
Email: {marc.sevaux,andre.rossi}@univ-ubs.fr

† University of Antwerp
Faculty of Applied Economics, Antwerp, Belgium
Email: kenneth.sorensen@ua.ac.be

1 Introduction

Mobile phones, music players, personal computers, set-top boxes and countless other digital electronics items are part of our daily life. These nomad devices offer more and more functionality. For example, recent mobile phones allow to communicate, to take pictures, to play video, to listen to music, to watch TV, to browse the WEB and so on. Thus, integrated circuits (IC) achieving all these functions become more and more complex.

Figure 1 shows the evolution of architectures (blue), standards (yellow) and throughput (red) over the last decade. Architectures, first based on System on Chip (SoC), went through Multi-Processors SoC, Network on Chip (NoC), Crypto-Processors and now Adaptive MP-Soc. For example, TV Standards at the end of the previous century were MPEG2 and MPEG4. Now the HD-TV is based on H264 standard. Furthermore, these new standards require increasing throughput, up to 1 GBit/s and more today.

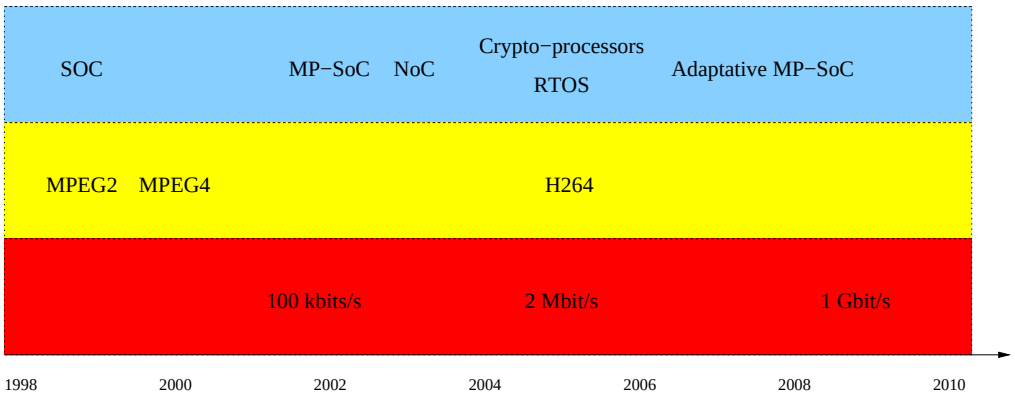


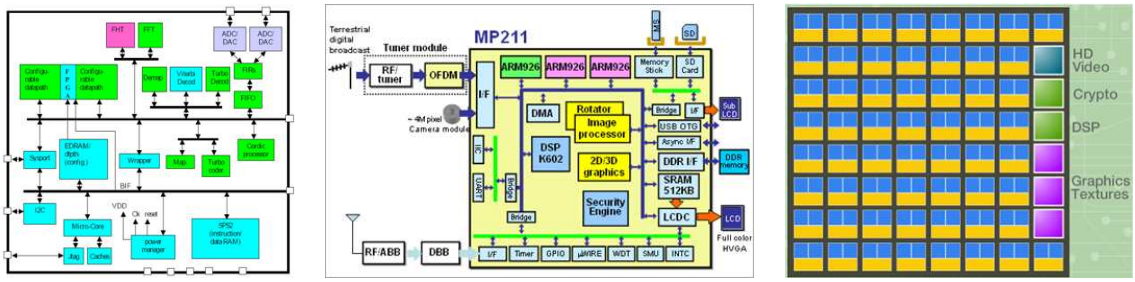
Figure 1: Evolution of architectures, standards and throughput

Designers aim at developing all these products taking into account several axes:

- Time to market reduction
- Integrated circuits miniaturization
- Low power consumption
- Throughput increase

This leads to a large increase in the global complexity of the problems and causes IC design methodologies to change. Of course, one can see that these objectives are conflicting and appropriate strategies should be used to deal with them smartly.

Concerning architectures complexity, FPGAs were a major evolution twenty years ago. A FPGA is a chip that can be configured as desired using libraries. The designer can decide which parts of the chip will be registers (memory banks), which part will be processors (or co-processors), and how they will be connected together. This is a big change compared to an ASIC which is a dedicated circuit that cannot be self-configured. Figure 2(a) shows a classical FPGA framework using a bus-hierarchy and IP-based designed. IP here refers to *Intellectual Properties* or bricks to be integrated to a global design.



(a) FPGA (2003) (b) MPSoc (2005) (c) TeraScale (2007)

Figure 2: Chip complexity

A Multi-Processor System on Chip, as in Figure 2(b), is a more complex structure where some processors are dedicated to some tasks and connected together. Those MP-SoC are often designed for a class of tasks like Digital Signal Processing. They include processors for audio and/or image processing, 2D and 3D images, cryptography, antenna transmissions, wireless protocols, etc. Figure 2(b) is the MP211 chipset from 2005.

Adaptative Multi-Processors System on Chip are available since in 2007. The main difference is that some parts are dedicated to certain tasks (as in a MP-SoC) but the rest of the architecture is programmable (like in a FPGA) but can be reconfigured on-line. This means that if a part of the processor fails or is unable to reach the requirements (typically real-time constraints satisfaction), the tasks normally processed by this part of the MP-SoC can be transfered to another part of the processor by a partial or complete reconfiguration. Figure 2(c) depicts the last processor TeraScale from Intel Corporation.

2 Electronics community and optimization

For a long time, designers have been using a wide variety of optimization techniques such as Integer Linear Programming, heuristics and metaheuristics. An observation of the use of these techniques over the last two years has led to the following conclusions: even if the electronics community know

these methods, there is a great need for more formal techniques, more appropriate models and efficient solution approaches tailored for specific problems in the electronics industry.

In the sequel, we will introduce several problems where the help from the metaheuristic community is needed and for which collaborations are necessary. This talk will be the starting point for the creation of a network of long term collaborations between metaheuristic and electronics communities.

Several applications in electronic design require optimization, to name a few:

- Solving data allocation to memory banks (graph coloring)
- Reducing latency in data transfer in a NoC - Network on Chip (Shortest Path Problems with constraints)
- Multiobjective optimization in High level Synthesis (multiobjective scheduling)
- Optimizing parameters in DSP filters (non-linear optimization)
- Reducing treatment under satellite norm data transfer (Hamiltonian path problems)
- Robustness and flexibility analysis for above problems

In the rest of the paper, we present some of these applications with their main characteristics and what has been done up to now, either by people from the electronics community or by us.

3 Potential applications

3.1 Network-based problems

Network-on-Chip (NoC) are electronic chips where several parts are considered as processors and a network between them feeds the processors with data and computations to run.

There are many common links with computer network based problems. For each NoC, synchronization, routing, security and reliability are also required. But at the same time, information that is sent through the NoC consists most of the time of repetitive tasks and the quantity of information is several gigabytes over a very short timespan (less than a few milliseconds). Communications have to be handled either in a centralized or decentralized manner as well as memory management.

One of the problem that interests electronic designers is latency reduction in data transfer in a NoC. This can be seen as a *Shortest Path Problems with constraints* in terms of operations research

Problem statement A Network on Chip (NoC) [2] is an internal structure of an electronic component that serves to transfer data and executes operations on distant processors. A graph represents the NoC with specific input and output ports. Data has to be exchanged between these ports. Of course, the shortest path between source and sink nodes is always the best path unless some arcs are occupied with other data. In that case, classical shortest path algorithms cannot be used anymore and should be replaced with different approaches for handling these constraints. The goal is to make an efficient implementation of such an algorithm taking into account the specific structure of the graph and of the constraints [1].

Main characteristics of the problem The number of data units to be transfered is very large (about 3/5 gigabytes of data). Fortunately the network itself is rather small (less than 100 nodes and $\approx$ 200 edges).

First steps of a general approach A first MILP model of this problem has been proposed, mainly for the sake of modeling it. The NoC has n nodes, it is modeled as a non-directed graph $G = (V, E)$. There are o packets to be transported by this NoC, each packet k having an origin and a destination, modeled as follows: for all node i in V, $o_{i,k} = 1$ if packet k has i as its origin, $d_{i,k} = 1$ if packet k has i as its destination. Edges are modeled with a matrix c defined by $c_{i,j} = 1$ is and only if node i and node j are connected (i.e. if $(i, j) \in E$). Time is discretized, and the problem is solved in a finite horizon H where H is assumed to be given. There are $n \times n \times o \times H + 1$ decision variables defined as follows: $x_{i,j,k,t} = 1$ if packet k moves from node i to node j at time t, and time T, that is the maximum arrival time over all the packets. The problem can be formulated as follows:

$$
(MILP): \begin{cases}
\text{Minimize } T & \\
\displaystyle\sum_{k\in[1,o]} x_{i,j,k,t} \leq 1 & \forall i \in [1,n], \forall j \in [1,n], \forall t \in [1,H] \quad (1)\\
x_{i,j,k,t} \leq c_{i,j} & \forall i \in [1,n], \forall j \in [1,n], \forall k \in [1,o], \forall t \in [1,H] \quad (2)\\
\displaystyle\sum_{\substack{t\in[1,H]\\ j\in[1,n]}} x_{i,j,k,t} = o_{i,k} & \forall i \in [1,n], \forall k \in [1,o] \quad (3)\\
\displaystyle\sum_{\substack{t\in[1,H]\\ i\in[1,n]}} x_{i,j,k,t} = d_{j,k} & \forall j \in [1,n], \forall k \in [1,o] \quad (4)\\
x_{i,j,k,\tau} \leq 1 - \displaystyle\sum_{\substack{t<\tau\\ h\in[1,n]}} x_{i,h,k,t} & \forall i \in [1,n], \forall j \in [1,n], \forall k \in [1,o] \quad (5)\\
\displaystyle\sum_{\substack{t\in[1,\tau-1]\\ l\in[1,n]}} t \times x_{l,i,k,t} \leq \sum_{j\in[1,n]} t \times x_{i,j,k,t} & \forall i \in [1,n], \forall k \in [1,o], \forall \tau \in [2,H] \quad (6)\\
\displaystyle\sum_{\substack{t\in[1,H]\\ i\in[1,n]}} t \times x_{i,j,k,t} \leq T & \forall k \in [1,o], j | d_{j,k} = 1 \quad (7)\\
T \geq 0, \quad x_{i,j,k,t} \in \{0,1\} & \forall i \in [1,n], \forall j \in [1,n], \forall k \in [1,o], \forall t \in [1,H]
\end{cases}
$$

Constraint (1) specifies that at any time, a packet can only move across one edge, a packet can only move across existing edges (2), each packet starts at its origin, i.e. packet k must leaves node i (its origin) to another node (say j) at some time instant t (3), each packet ends at its destination, i.e. packet k must reach node j (its destination) from another node (say i) at some time instant t (4), a packet k cannot move across edge (i, j) at time τ if it left node i earlier (at any time t such that $t < \tau$) (5), a packet k cannot move across edge (i, j) at time τ if it did not reach node i at some time t in $[1, \tau - 1]$ (6) and each packet k reaches its destination before time T (7). The goal is to minimize the total time T.

Of course, this approach cannot be practically used since it involves too many variables as it grows with o, the number of packets. So it allows space for applications of several heuristics and metaheuristics.

3.2 Sizing and assignment problem

How many memory banks are needed, and once defined, in which of these should data be placed to minimize the number of remaining conflicts while reading the data? That is the type of question a designer can ask when designing an architecture. This problem can be solved using variants of the *graph coloring problem*.

Problem statement In electronic design, several tasks have to be performed in order to obtain an architecture that meets the needs of the final consumer. That goal is reached after a series of synthesis and modelling steps. One of these steps concerns the data memory allocation. The designer has to

decide in which memory banks data will be stored in order to minimize access conflicts (two pieces of data cannot be accessed at the same time if they are in the same memory bank and this implies a delay in data transmission). In an early work, we have modelled the problem as a variant of the graph coloring problem where the nodes of the conflict graph have to be colored in such a way that two adjacent nodes have different colors (colors represent the memory banks). The goal is to extend this model to take into account specific constraints of the electronic application and find the best solution method. Two kinds of problems can be stated. Either, the number of memory banks has to be determined (without adding a delay in data processing) and the problem to be solved is the classical graph coloring problem, or the number of memory banks is fixed and we have to find a coloring that minimize the weights of the conflicting edges [4].

Main characteristics of the problem Instances for this kind of problems come mainly from electronic applications (e.g. MPEG or HD-TV encoder/decoder). This results in huge amount of source code written in C or C++. The size of the conflict graph can count up to several thousands of nodes and have a high density. This kind of instances is know to be easy for the classical problem, but for the k-weighted graph coloring problem, it is not the case.

First steps of a general approach Up to now, we have again modeled the problem as a MILP and used it for resolution:

$$\begin{cases} Minimize \quad \sum_{k=1}^{o} d_k.y_k & \\ \sum_{j=1}^{m} x(i,j) = 1 & (\forall i \in [1,n]) \\ x(k1,j) + x(k2,j) \leq 1 + y_k & (\forall j \in [1,m])(\forall k \in [1,o]) \\ x(i,j) \in \{0,1\} & (\forall i \in [1,n])(\forall j \in [1,m]) \\ y_k \in \{0,1\} & (\forall k \in [1,o]) \end{cases}$$

The objective is to minimize the weights of the conflicting edges, the first constraint forces each node to have a single color and the second constraint fixes the value of y_k for a conflicting edge.

For this problem, a PhD program has been started recently and the student has already produced interesting theoretical results (some complexity results, a new upper bound on the number of colors and some new properties on conflicting edges). Now the work is extended in the metaheuristics direction by deriving classical approaches of the graph coloring problem for the k-weighted graph coloring problem.

A new extension is also studied: a set of n data structures $\{a_1, a_2, ..., a_n\}$ has to be allocated to a RAM page b_j (j is in $[1,m]$ as there are m pages available). The data structure a_i has a size s_i, and the page b_j has a capacity c_j. Moreover, there are o pairs of data structures that should be placed in the same page. A pair p_k is defined by $p_k = \{a_{k1}, a_{k1}, d_k\}$, where a_{k1} and a_{k2} are two data structures, and d_k is the pair cost. Each pair has also a status: it is OFF if the two data structures share the same page, it is ON otherwise. If a pair is ON, the cost d_k should be paid. If it is OFF, then no cost should be paid. The purpose of this problem is to allocate one page to each structure, while minimizing the total cost.

3.3 Multiobjective optimization

In High level Synthesis, we aim at solving at the same time, resource allocation, data assignment and scheduling sub problems for different criteria [5]. This leads to a *multiobjective resource scheduling problem.*

Problem statement High Level Synthesis (HLS) is a modelling phase of electronic design which consists in allocating resources, scheduling operations and binding data into memory. During the scheduling phase, it is possible to find starting time of operations in order to minimize latency (total duration of the execution). It appears that a very good schedule for this objective can be rather poor in power consumption, in silicium (silicon) surface or in other criteria. Using a method based on an estimator for the solution cost, we are able to find different solutions and measure their value for the different objectives. We aim to implement a multiobjective optimization method to find the best trade-off between these solutions and let the designer choosing the best solution [3].

Main characteristics of the problem The precedence graph of the operations, which is the main input data of our problem counts for some instances more than 45 000 nodes (to be compared with the ≈200 tasks scheduling problems generally dealt in production). This is challenging, even just for handling data.

Steps of a general approach We are currently concluding a thesis on that general problem where a VNS is returning very competitive results and will be coupled with a GRASP approach. But up to now, a linear combination of the objectives is taken into account. Moreover the power consumption criteria is still not addressed in this work. The VNS algorithm improves initial solutions up to 67% and is always superior to practitioners ad-hoc heuristics as "left edge". However, "modified left edge" algorithms sometimes performs better than VNS but this happens for instances that have been specifically designed for MLE.

4 Conclusion and future work

This paper has aimed at presenting optimization problems raised by electronic practitioners and designers and aimed at attracting new partners for long term collaboration. It has been decided, as a strategic plan for our lab, to start common PhD programs with interested people and make efforts with the metaheuristic community to model and solve these problems more efficiently. In a mid-term horizon, an European project should be started on those topics as well.

References

[1] R. Dafali, J-Ph. Diguet, and M. Sevaux. Key research issues for reconfigurable network-on-chip. In *Proceedings of the International Conference on ReConFigurable Computing and FPGAs, ReConFig'08*, pages 181–186, Cancun, Mexico, 3-5 December 2008.

[2] S. Evain, R. Dafali, J-Ph. Diguet, Y. Eustache, and E. Juin. μSpider cad tool: case study of NoC IP generation for FPGA. In *Proceedings of conference DASIP'07*, France, 2007.

[3] D.D. Gajski, N.D. Dutt, A.C-H. Wu, and S.Y-L. Lin. *High-level synthesis: introduction to chip and system design*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[4] T.R. Jensen and B. Toft. *Graph coloring problems*. Discrete Mathematics Optimization. Wiley-Interscience, New York, USA, 1994.

[5] B.M. Pangrle. On the complexity of connectivity binding. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10, 1991.